10-Sept-85

For Developers using the IBM cross development environment:
==========

* * * * NEWS * * * *

The Amiga should now be in your local computer stores. Some software is already available and other packages are not far off. This is an exciting time for us all. Thank you once again for supporting our machine.

1) Read has been rewritten and should now work both ser and par.
   Changes:
   a) must start READ before sending to it
   b) ^C can be used to exit
   c) 9600 baud is beleived to be relyable
   d) Any length file can be sent (if you've got the disk space).
   e) You need to use the new convert program in order to generate files (transmissions) without line-feeds.
   f) if you STILL have problems with read over running try "read ram:foo serial".

2) The diskette organization has changed (once again?). There are 3 C disks and 2 Assem disks.

3) Linker supports overlays. Reassemble ovs.asm AND rename if you need more than 20 overlay tasks. Read more about overlays in the back of the DOS developers manual. Tasks must be added 20 at a time (20, 40, 60...) and the file renamed to ovs.040, ...

4) Startup program names now match Amiga development environment.

5) The new AmigaDOS diskvalidator is NOT compatible with the old ones. Furthermore the old ones are NOT compattible with the new DOS. You should remove any old ones.

Equipment Requirements:
==========

IBM (or similar) equipment required:

IBM personal computer
512 or 640k recommended
1 360k 5-1/4" floppy diskette
DOS 2.0, 2.1, 3.0, or 3.1
Either an additional floppy diskette or hard disk.
(trying to use 2 floppies is NOT recommended!)
Monitor
A serial port. Connected to the Amiga.

Amiga equipment required:

Amiga system (CPU, mouse, keyboard) V25 or later.
Monitor. Sony CDP-1201, set to Analog, preferred.

Amiga V 1 KickStart diskette
Amiga V 1 CLI (DOS) or Workbench diskette.

Diskette Contents:
==================

This developer's package contains the following IBM format diskettes:

Progm.Tools	- C compiler + other tools
Include.Fls	- C include files (lots of them)
Library.Tls	- C startup files and libraries and additional tools.

These diskettes are formatted as double sided with 9 sectors per track. They require IBM or MS DOS 2.0 or better to be read and used.

>>==========<<
The following "features" are known to exist with the included programs:
>>==========<<

Alink (on the IBM):

There is an unconfirmed report (from a reliable source) that the linker will "blow-up" in pass 2 if you include a .OBJ file explicitly that also exists in a library. Try not to do this. This is still under investigation.

If you create a library of assembly object modules that were created with the -S option you may find that link will try to include ALL of the library... not just the requested modules.

"With" files are very sensitive. You may NOT have blank lines or <Tab> characters in it. The last character (or characters) must be MS-DOS end of file marks (^Z).

Given the overlay description:

```
from a                    a
overlay                   |
b                  b------c------g
c                         |
* d               d---------e
* e                        |
** f                       f
g
#
```

```
a can call b
b can call a      but NOT c,d,e,f,g
c can call a,d,e but NOT f
d can call a,c
e can call a,c,f
f can call a,c,e but NOT d,b,g
```

Any attempt to call an overlay that you are not allowed to may result in the misleading error message: "Invalid object type 6 in file foo".

>>==========<<

Assem (on the IBM):

No known problems. Be sure your 1st "SECTION" directive is before your include directives. Failure to do this may cause the loader to start your program in some place other than the code section.

Well the loader is getting smarter and will try to avoid executing zero length hunks even if they are the 1st to load.

>>==========<<

Lattice C "features":

This version of C has been upgraded to the large memory model. The size or the C programs you can compile should now be quite large. For example if you try to include all of the "*.H" files at once the compiler will process this correctly.

The code generation for REG (register) type variables has been fixed!!
The code generation for backwards goto's has been fixed!!
%g in printfs does not work well, use %e or %f.
%*d is not available (and may never be)
%X, the same.

>>==========<<

C Runtime support:

The return code (exit value) is now being passed back to AmigaDOS. Trying to use FAILAT with your C programs should work. Well maybe it always returns a 1. Were looking at this.

printf ("%C", 255) will NOW display the right character.
printf now buffers.

>>==========<<
Lattice C notes.
>>==========<<

You should read:
The manual "Lattice C Compiler for the 8086/8088 Series Microprocessor" as a background for the other documentation. In particular this documents error codes and an overview of the C language as implemented.

In addition you should read the documentation provided whose header is "Lattice 68000 C Compiler". Ignore information relative to Lattice or Quello object formats. The C provided produces Amiga Binary object format.

Differences from other Lattice C's:

The type char is signed.
The new types 'unsigned char' and 'unsigned long' have been introduced.

By default comments are not considered to be nested.

There is no -a option. The algorithm for pointer aliasing is different from that described.

This C should be a full implementation of C compatible with the Unix (a trademark of Bell Labs) system V, C compiler. It includes such features as 'enum' types and the ability to return structures.

>>==========<<
Convert notes:

Convert takes a binary input file (the output of alink) from its standard input (<) and writes a text file of ASCII digits representing the hexadecimal value of its input terminated by a Q to the standard output file (>). It no longer inserts <new-line>s in the file.

>>==========<<
Linker/Library notes:

An Amiga library, by definition, is the concatination of one or more object files. To create your own library on the PC:

1) Create files (libent1.obj, libent2.obj, ..)
2) Concatinate them all together. The /B option is required!!!
   (copy /B libent1.obj+libent2.obj mylib.lib)

>>==<<
How to use the development environment.

The examples and batch files on these diskettes assume the following configuration:

A: Floppy, able to read 360k diskettes
B: not used
C: Hard disk (or similar). Requires 1.6 meg for the files on these diskettes. An additional 200k (minimum) should be left for your source code and work space.
D: - Z: not used.

Every attempt has been made to make the hard disk label (C:) as independent as possible. You should only need to:

a) Have your hard disk as your default volume. -- and --
b) Change "C:" to "D:" (or whatever) in the MAKEHD.BAT batch file in order to use another disk drive.

>>==<<
For all programmers:
>>==<<

If you boot with a workbench diskette you will need to:

1. Run SETPREFS and select CLI = ON.
2. Double click the cli (1>) Icon to open a cli window.
3. Development then proceeds "normally" within the CLI environment.

I suggest that you run your programs surrounded by "AVAIL" commands. Make sure you (and C and the ROM code) are freeing up all the memory you've used. Some operations (like putting a "new" diskette in a drive) can allocate a bit more memory on a more or less permanent basis. But in most cases memory use should remain constant. It is also possible to load a disk based library (speech, printer, ...) which will occupy space until that space is required. The sequence: avail/prog/avail2/prog/avail3 should result in avail2 matching avail3 if this is the case.

The file LSTARTUP.OBJ works with the Lattice run-time system (LC.LIB). The file STARTUP.OBJ will work if you do not need to link with LC.LIB. It can be used for both CLI and Workbench applications. See other dosumentation.
"WBSTARTUP.OBJ" is no longer necessary, use STARTUP.OBJ.

>>==<<
Development for C programmers:
>>==<<

```
(create files file1.c and file2.c)
lc1m file1
lc1m file2
lc2m file1
lc2m file2
alink \lib\startup.obj+file1.o+file2.o library=\lib\lc.lib+\lib\amiga.lib
      to file.ld
```

You may use the "-s" option on the (pass 2) command line to prevent the default "scatter loading" of your code and data. This is useful for debuging, because all of your code will be loaded in one place. It is

not "system-friendly" In that it requires larger chunks of memory for your program to load. Please do NOT ship any software compiled this way!

See also the MAKE???.BAT and README files in \DOS on the LIB\Examples diskette.

>>==========<<

For Assembly programmers:

>>==========<<

```
(create files file1.asm and file2.asm)
assem file1.asm -o file1.obj -l file1.lis
assem file2.asm -o file2.obj -l file2.lis
alink file1.obj+file2.obj library=/lib/amiga.lib to file.ld
```

To get the program to the Amiga, connect your IBM serial port to the Amiga serial port. Then assuming AUX is the name of the serial port on you IBM, the following should be typed:

```
Amiga:   1> READ file SERIAL
IBM-PC:  A> convert <file.ld >AUX
IBM-PC:  A> dir file.ld
Amiga:   1> list file
```

Compare the file size on the IBM with that on Amiga. The sizes should be the same if a successful transfer has been made.

If this does not work for you please try the following:

```
Amiga:   1> READ file SERIAL
IBM-PC:  A> convert <file.ld >file.out
IBM-PC:  A>    ... run any "normal" telecommunications program (e.g.
               PC-TALK) to send the .OUT file using your serial port.
IBM-PC:  A> dir file.ld
Amiga:   1> list file
```

Then to run the program

```
Amiga:   1> file
```

Assembly programmers may not need to use /lib/amiga.lib if they don't have any external references to Amiga provided values.

>>==========<<

Additional notes:

If for some reason you "upload" a .ld (or any binary) file from the IBM PC to the Sun you will have to use the "adjust" program to trim off any extra "data" in the last block. Use the file size as determined by a DIR on the PC. The format of the command is:

```
adjust file_name length_in_bytes
```

>>==========<<

Final note:

Thank you for your support. We can't do it alone, but with our machine and your software we'll knock their socks off!

Good luck.

V 1.0 Macro Assembler Header Problem 25-Sept-1985

Many of the Macro Assembler include files contain ENDC directives of the form: ENDC argument (example: ENDC EXEC_IO_I).

One of our programmers has found that this generates many assembler errors. By removing the arguments from the ENDC directives, he was able to assemble his code.

If you are only including the following:

exec/types.i
intuition/intuition.i

then you must correct the ENDC directives in these files:

exec/libraries.i	lines 19, 125
exec/nodes.i	line 50
exec/lists.i	lines 19, 144
exec/ports.i	lines 19, 23, 68
exec/types.i	line 117
exec/io.i	lines 19, 23, 127
devices/timer.i	line 24

If you are including additional headers, you must correct any ENDC directives with arguments in those headers and any others that they include.

ENDC directives with arguments can be found in most of the header files in these include subdirectories:

exec, devices, resources, libraries, workbench, hardware, diag

You can find the file names and line numbers by using search.

example: search df1:include/exec ENDC

Use space bar to pause the search, return key to resume.
Watch for ENDC followed by an argument.